Lecture 19

· Review Session Friday Nov 17 4-6

HW3 due Tues by noon (12 hr extension!)

HW3

Q2: show (i) $\Rightarrow$ (ii) + (ii) $\Rightarrow$ (i)

To show (i) $\Rightarrow$ (ii) [show if $\exists$ an onto fxn $g: \mathbb{N} \to S$ then $\exists$ 1-1 fxn $f: S \to \mathbb{N}$]

3 steps

[① Define f

[② show your f is well-defined
($\forall s \in S$, $f(s)$ = one element in $\mathbb{N}$)

[③ show f is 1-1

Recap from Last week

① $D = \{ \langle M \rangle \mid M(\langle M \rangle) \text{ does not accept} \}$ is not r.e.

Diagonal Language

Proof by diagonalization

② $\overline{D} = \{ \langle M \rangle \mid M(\langle M \rangle) \text{ halts and accepts} \}$

is r.e. but not recursive

($\forall L \subseteq \Sigma^*$ If L recursive then $\overline{L}$ also recursive)

③ $A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

is r.e. but not recursive

$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

D not r.e.
$\overline{D}$ is re, no rec
A_{TM} is re, not recursive

- We saw that A_{TM} is r.e. / recognizable.

Pf that A_{TM} is not decidable:

Assume for sake of contradiction there is a decider N for A_{TM}.
We will use M to construct a decider N' for $\tilde{D}$:

N':

On input $\langle M\rangle$:
check if input is a legal encoding of a TM. If not reject
Otherwise Run N on $\langle M, \langle M\rangle\rangle$
If N accepts $\longrightarrow$ accept
If N rejects $\longrightarrow$ reject

Since N always halts, N' always halts.
Also N' accepts $\tilde{D}$. Contradiction since $\tilde{D}$ is not decidable
$\therefore A_{TM}$ is not decidable

TM Reductions

The previous proof showing that A_{TM} is not decidable is a reduction; we showed:

$$\text{a decider for } A_{TM} \implies \text{a decider for } \tilde{D}$$

Since $\tilde{D}$ not decidable, A_{TM} also not decidable.

So we showed $\tilde{D} \leq_T A_{TM}$

Defn Language A is TM-reducible to Language B, written $A \leq_T B$ if a decider for $B \implies$ a decider for A

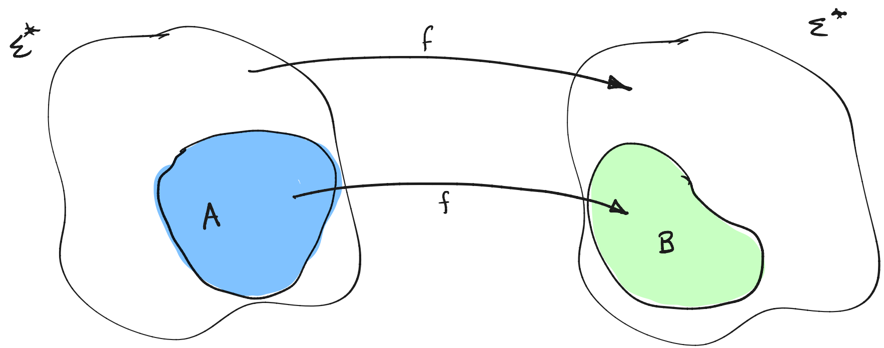
Important If $A \leq_T B$ and B is decidable then A is decidable

If $A \leq_T B$ and A is not decidable, then B is not decidable (contrapositive)

More on Reducibilities

A language A is mapping-reducible to language B ($A \leq_m B$) if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$\forall x \in \Sigma^* \ (x \in A \iff f(x) \in B)$$

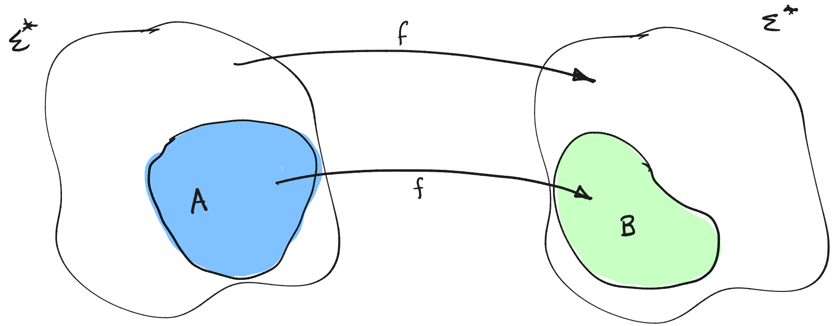


- f maps strings in A to strings in B, and strings not in A to strings not in B

More on Reducibilities

A language A is mapping-reducible to language B ($A \le_m B$) if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$\forall x \in \Sigma^* \ \left(x \in A \iff f(x) \in B \right)$$



- f maps strings in A to strings in B, and strings not in A to strings not in B

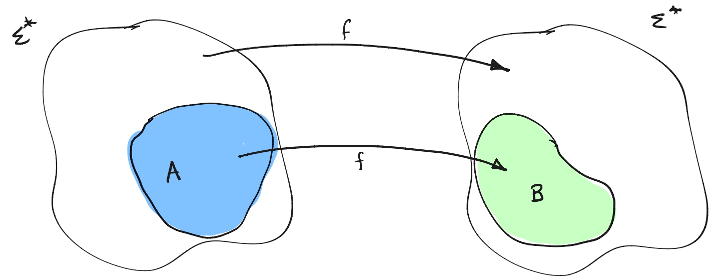
Lemma If $A \le_M B$ then $A \le_T B$

(mapping reductions are special case of Turing Reductions)

More on Reducibilities

A language A is mapping-reducible to language B ($A \leq_m B$) if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$\forall x \in \Sigma^* \left(x \in A \iff f(x) \in B \right)$$



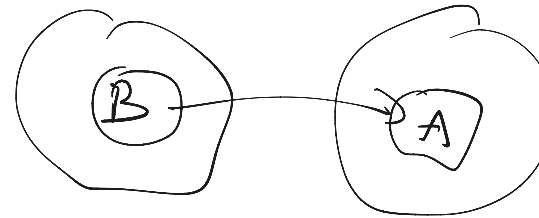
Lemma Let $A \leq_m B$. Then:

① B decidable $\Rightarrow$ A decidable (or equivalently, A undecidable $\rightarrow$ B undecidable)

② B recognizable/re $\Rightarrow$ A recognizable/r.e.

Lemma If we have $A \leq_m B$, then we also have $\bar{A} \leq_m \bar{B}$

$B \leq_m A$:



Q: To show A is not decidable

Do I want to show

(1) $A \leq_m B$ for some B that is undecidable

or (2) $B \leq_m A$ for some B that is undecidable

correct answer

$B \leq_m A$ means

(2): If A is decidable then B is decidable

So B not dec $\rightarrow$ A not dec. ✓

Example : HALT = $\{ \langle M, x \rangle \mid M$ halts on input $x \}$

Lemma 1 HALT is r.e. (exercise)

Lemma 2 Halt is not recursive/decidable

Proof of Lemma 2 We will show $A_{TM} \leq_T$ HALT

Then since A_{TM} not decidable, this implies HALT not decidable.

Let N be an (alleged) decider for HALT. We will use N to create a decider, N' for A_{TM}

N' :

on input $\langle M, x \rangle$:
check if input is legal encoding of a TM M, followed by x (halt if not)
Run N on input $\langle M, x \rangle$
If N accepts, simulate M on x. Accept $\langle M, x \rangle$ if simulation accepts; otherwise reject $\langle M, x \rangle$
If N rejects, halt and reject

Example : HALT = $\{\langle M, x\rangle \mid M$ halts on input $x\}$

Lemma 1 HALT is r.e. (exercise)

Lemma 2 Halt is not recursive/decidable

Proof of Lemma 2 We will show $A_{TM} \le_T$ HALT.
Then since A_{TM} not decidable, this implies HALT not decidable.
Let N be an (alleged) decider for HALT. We will use N to create a decider, N′ for A_{TM}

Note:
This is a Turing reduction but not a mapping reduction

N′ :

> on input $\langle M, x\rangle$:
> check if input is legal encoding of a TM M, followed by x (halt if not)
> Run N on input $\langle M, x\rangle$
> If N accepts, simulate M on x. Accept $\langle M, x\rangle$ if simulation accepts; otherwise reject $\langle M, x\rangle$
> If N rejects, halt and reject

Proof of correctness: First, if N is a decider for HALT then N′ will halt on all inputs.
Now for correctness: First if M halts on x, then N′ just simulates M on x and does the same thing, so N′ will also halt + accept $\langle M, x\rangle$
otherwise if M does not halt on x, then N will not accept so N′ will also halt and reject.

<u>Example</u> : HALT $= \{ \langle M, x \rangle \mid M$ halts on input $x \}$

<u>Lemma 1</u> HALT is r.e. (exercise)

<u>Lemma 2</u> Halt is Not recursive

<u>Lemma 3</u> $\overline{\text{HALT}}$ is Not r.e.

If Halt, $\overline{\text{HALT}}$ both re, then HALT would be decidable (By closure property). ∴ By Lemma 2, $\overline{\text{HALT}}$ Not r.e.

<u>Closure Prop:</u>

(If L and $\overline{L}$ are both r.e., then they are both recursive.)

$\overline{\text{HALT}} = \{ \langle M, x \rangle \mid M$ does not halt on input $x \}$

<u>Example:</u> Nonempty = $\{ \langle M \rangle \mid M$ accepts at least one string$\}$

i.e. $\mathcal{L}(M)$ is not empty

① Nonempty is r.e. (Pf: use dovetailing)

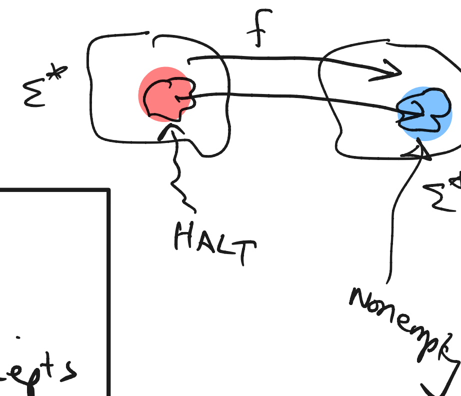
Example Nonempty = { <M> | M accepts at least one string }

① Nonempty is r.e. (Pf : use dovetailing)

② Nonempty is Not recursive/decidable.

Assume for sake of contradiction N is a decider for Nonempty.

We will use N to construct a decider, N' for HALT



N' :

on input $\langle M, x \rangle$: Let M' be a TM that on input w, M' ignores its input and simulates M on x. If M halts on x then M' halts and accepts Run N on $\langle M' \rangle$ If N accepts $\langle M' \rangle$ $\rightarrow$ halt and accept otherwise $\rightarrow$ halt and reject

$f : \langle M, x \rangle \rightarrow \langle M' \rangle$

* M' depend on M + x.

Example Nonempty = $\{ \langle M \rangle \mid M$ accepts at least one string $\}$

(1) Nonempty is r.e. (Pf: use dovetailing)

(2) Nonempty is Not recursive/decidable.

Assume for sake of contradiction N is a decider for HALT

We will use N to construct a decider, N' for A_{TM}

N':

on input $\langle M, x \rangle$: Let M' be a TM that on input w, M' ignores its input and simulates M on x. If M accepts x then M' halts and accepts Run N on $\langle M' \rangle$ If N accepts $\langle M' \rangle \longrightarrow$ halt and accept otherwise $\longrightarrow$ halt and reject

M' accepts all strings if M halts on x

M' accepts NO strings if M does not halt on x

$\mathcal{L}(M')$ Nonempty iff M accepts x

Note: This is a mapping reduction showing $A_{TM} \leq_m$ Nonempty:

$f : \langle M, x \rangle \longrightarrow \langle M'_{M,x} \rangle$

Example Nonempty = { <M> | M accepts at least one string }

1. Nonempty is r.e. (Pf: use dovetailing)
2. Nonempty is Not recursive/decidable.
3. $\overline{\text{Nonempty}}$ = EMPTY = { <M> | $\mathcal{L}(M) = \phi$ }

   EMPTY is Not r.e. since if it were r.e., then NONEMPTY would be recursive

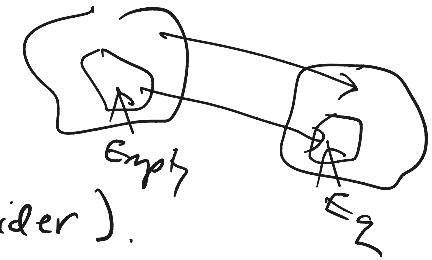
(For any $L \subseteq \Sigma^*$, If L and $\bar{L}$ are both r.e. then both are recursive)

L is recursive $\Leftrightarrow$ $\bar{L}$ is recursive

Summary so far

1. D is not r.e.
2. $\overline{D}$ is r.e. but not recursive
3. A_{TM} is r.e. but not recursive
4. Halt is r.e. but not recursive, $\overline{\text{Halt}}$ is not r.e.
5. Nonempty is r.e. but not recursive, Empty is not r.e.

Example: $EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$



Claim EQ_{TM} is not recognizable (not r.e.)

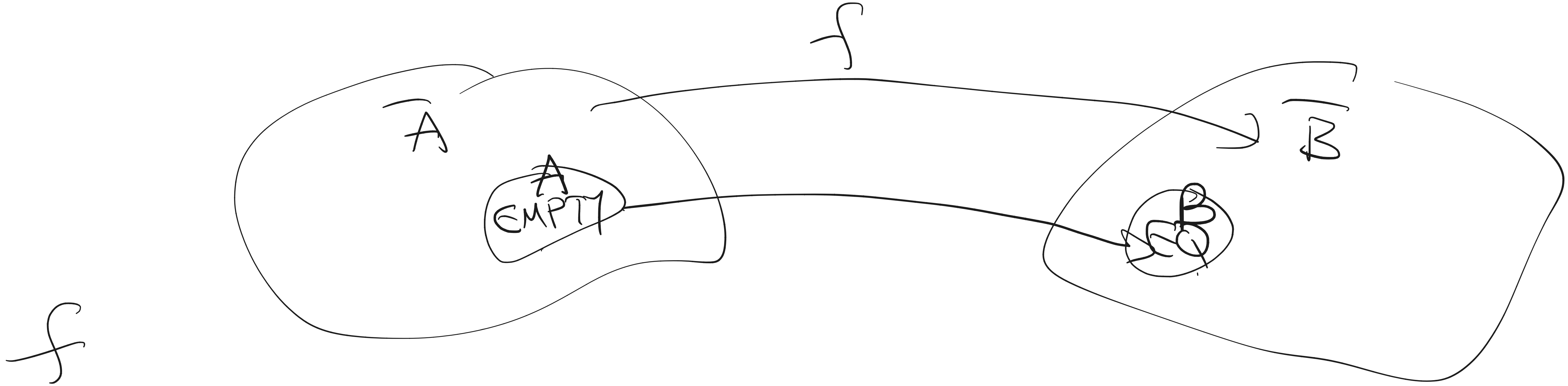
Pf: We will show $EMPTY \leq EQ$

Let N be a TM for EQ_{TM} (not necessarily a decider).
We will construct a TM N' for EMPTY as follows:

N':

on input $\langle M\rangle$:
Run N on input $\langle M, M^{\phi}\rangle$ where M^{ϕ} is a TM that rejects all inputs.
If N halts and accepts $\langle M, M^{\phi}\rangle$ then accept $\langle M\rangle$ otherwise if N halts and rejects $\langle M, M^{\phi}\rangle$ then reject $\langle M\rangle$

$$f : \langle M\rangle \longrightarrow \langle M, M^{\phi}\rangle$$



If EQ is recursive $\rightarrow$ empty is recursive

If EQ is r.e. $\rightarrow$ empty is r.e.

given w : Is $w \in$ empty?

compute $f(w)$

Then if EQ is recursive

Run TM for EQ on $f(w)$

accept iff TM for EQ accepts

Example: $EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$

What about $\overline{EQ_{TM}}$?

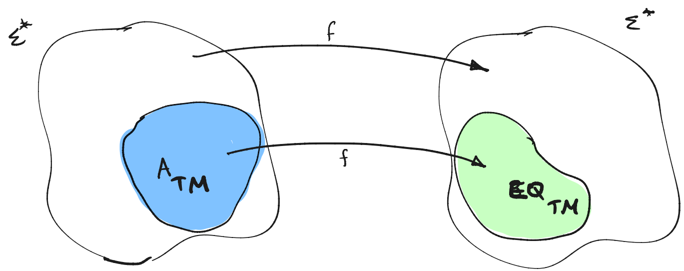
goal: map

$\langle M, x\rangle \longrightarrow \overbrace{\langle M', M''\rangle}$

s.t. M accepts x iff $\mathcal{L}(M') = \mathcal{L}(M'')$

Claim $\overline{EQ_{TM}}$ is not r.e.

Pf we will show: $A_{TM} \leq_m EQ_{TM}$

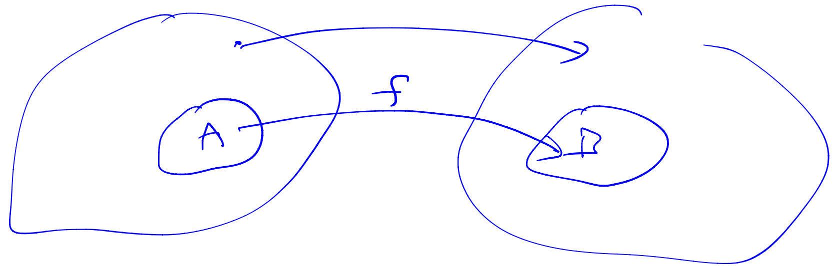


This is same as constructing a mapping reduction $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

Therefore since $\overline{A_{TM}}$ is Not r.e., $\overline{EQ_{TM}}$ is also Not r.e.

$A_{TM} = \{\langle M, x\rangle \mid M \text{ accepts } x\}$

$\overline{A_{TM}} = \{\langle M, x\rangle \mid M \text{ doesnt accept } x\}$

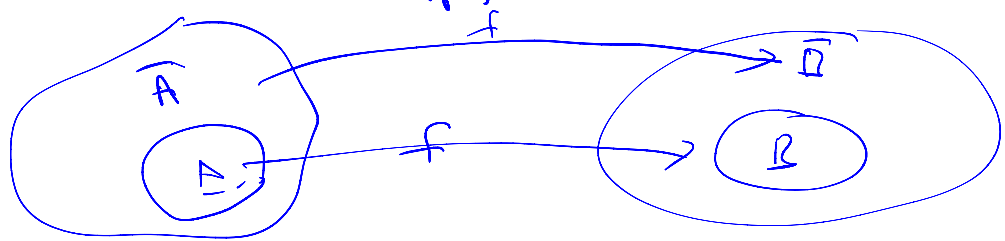


$\overline{B} = \overline{EQ}$

$\overline{A} = \overline{A_{TM}}$

say we have f mapping reduction from A to B.
so f maps yes instances of A to yes instances of B
and NO " " " to NO instances of B.

Note the same f is a mapping reduction from $\overline{A}$ to $\overline{B}$

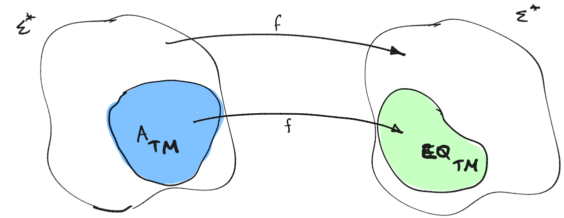


So using f it follows that if $\overline{B}$ is r.e.,
then $\overline{A}$ is r.e.

$\therefore$ by contrapositive, if $\overline{A}$ is not r.e. then we will have shown that $\overline{B}$ is not r.e.

Example: $EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$

What about $\overline{EQ_{TM}}$?



Pf We will show $A_{TM} \leq_m EQ_{TM}$

INCORRECT MAPPING REDUCTION:

f:

On input $s = \langle M, w\rangle$ Construct M_2 : on input x run M on w, accept x iff M accepts w Let $f(s) = \langle M, M_2\rangle$

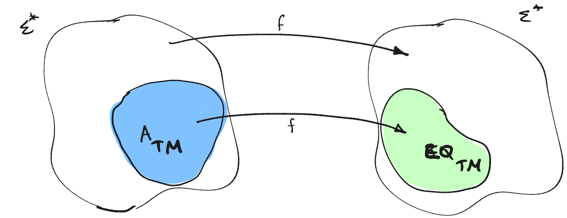
incorrect since $\mathcal{L}(M_2)$ is either Σ^* or ϕ

But we dont know anything about $\mathcal{L}(M)$. So could have M accepting w (so $\langle M, w\rangle \in A_{TM}$) but $\mathcal{L}(M) \neq \Sigma^*$

In this case f maps $\langle M, w\rangle \in A_{TM}$ to a pair $\langle M, M_2\rangle \notin EQ_{TM}$

Example: $EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$

What about $\overline{EQ_{TM}}$?



Pf We will show $A_{TM} \le_m EQ_{TM}$

Let M^{ALL} be a TM that accepts every input

f: $\boxed{\begin{array}{l} \text{On input } s = \langle M, w\rangle \\ \text{If } s \text{ not of the correct form reject} \\ \text{Else say } s = \langle M, w\rangle \\ \quad \text{Construct } M_2 : \text{on input } x \\ \quad\quad \text{run } M \text{ on } w, \text{ accept } x \text{ iff } M \text{ accepts } w \\ \quad \text{Let } f(s) = \langle M^{ALL}, M_2\rangle \end{array}}$

This is correct

since $\langle M, w\rangle \in A_{TM}$

iff $f(\langle M, w\rangle) \in EQ_{TM}$

$$f : \langle M, w\rangle \longrightarrow \langle \underbrace{M'}_{M^{ALL}}, \underbrace{M''}_{M_2} \rangle$$

Summary so far

① D is not r.e.

② $\overline{D}$ is r.e. but not recursive

③ A_{TM} is r.e. but not recursive

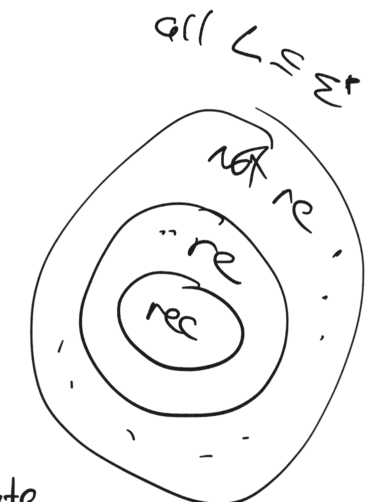
④ Halt is r.e. but not recursive, $\overline{\text{Halt}}$ is not r.e.

⑤ Nonempty is r.e. but not recursive, Empty is not r.e.

⑥ EQ_{TM} is not r.e., $\overline{EQ_{TM}}$ not r.e.

Tips for characterizing a given Language as (1) recursive; (2) recursive but not re; (3) Not r.e.

① Try obvious algorithms to see if you think L is recursive / r.e. (dovetailing technique useful to show r.e.)
Watch out for tricks -- if L defined based on some property of the machine (& not a property of L)



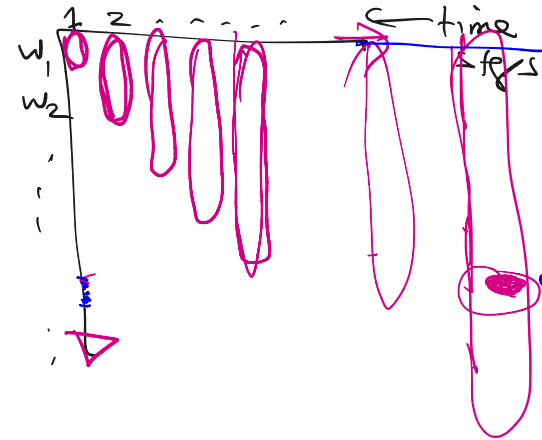
② To prove L is not r.e., sometimes helpful to look at $\bar{L}$
(If $\bar{L}$ is r.e. but not recursive then L not r.e.)

③ get reduction in correct direction!

④ Sometimes in reduction, need to construct an intermediate TM that ignores its own input.

Nonempty is R.E :

Nonempty = { <M> | M accepts at least one string }



Assume $\Sigma = \{0,1\}$
Enumerate all string over 0, 1

$w_1\ w_2\ w_3\ w_4$ - - - - -

TM for Nonempty :

On input <M>:

- For $i = 1, 2, 3, \ldots$
  - For $j = 1, 2, \ldots, i$
    - Run M on w_j for i steps
    - If M halts and accepts within these i steps
      - halt and accept